# Practicing Continuous Integration and Continuous Delivery on AWS

Accelerating Software Delivery with DevOps

By: Anand Prakash and Khadar Basha

White Paper

## Table of Contents

## Continuous Integration

Continuous Integration (CI) is a development methodology that involves frequent integration of code into a shared repository. The integration may occur several times a day, verified by automated test cases and a build sequence. It should be kept in mind that automated testing is not typically mandatory for CI. It is only practiced for ensuring a bug-free code.

### Benefits of Continuous Integration

Here are few benefits that have made continuous integration essential to any application development lifecycle.

- **Early Bug Detection**: If there is an error in the local version of the code that has not been checked previously, a build failure occurs at an early stage. Before proceeding further, the developer must fix the error. This also benefits the QA team since they mostly work on builds that are stable and error-free.
- **Reduced Bug Count**: In any application development lifecycle, bugs are likely to occur. However, with Continuous Integration and Continuous Delivery (CD0 being used, the number of bugs is reduced significantly, although it depends on the effectiveness of the automated testing scripts. Overall, risk is reduced a lot since bugs are now easier to detect and fix early.
- **Automated Process**: Manual effort is also reduced a lot since CI automates build, sanity, and a few other tests. This makes sure the path is clear for a successful continuous delivery process.
- **Process Transparency**: A great level of transparency is required in the overall quality analysis and development process. The team gets a clear idea when a test fails, what is causing the failure, and whether there are any significant defects. This enables the team to make a real-time decision on where and how efficiency can be improved.
- **Cost-Effective Projects**: Since the bug count is low, manual testing time is greatly reduced, and this clarity increases across the overall system. This optimizes the budget of the entire project.

## Continuous Delivery

Continuous Delivery (CD) is the process of getting all kinds of changes to production. Changes may include configuration changes, new features, error fixes, etc. They are delivered to the user in a safe, quick, and consistent manner.

The goal of Continuous Delivery is to make deployment predictable and scheduled in a routine manner. It is achieved by ensuring that the code always remains in a state where it can be deployed whenever demanded, even when an entire team of developers is constantly making changes to it. Unlike continuous integration, testing and integrating phases are eliminated, and the traditional process of code freeze is followed.

### Benefits of Continuous Delivery

If the best practices are followed, continuous delivery can help your application development in quite a few ways.

- **Reduced Risk**: The main goal of Continuous Delivery is to make deployment easier and faster. Patterns like blue-green deployment make it possible to deploy the code at very low risk and almost no downtime, making deployment totally undetectable to the users.
- **High-Quality Application**: Most of the process is automated, testers now have a lot of time to focus on important testing phases like exploratory, usability, security, and performance testing. These activities can now be continuously performed during the delivery process, ensuring a higher quality application.
- **Reduced Cost**: When an investment is made on testing, build, and deployment, the product evolves often throughout its lifetime. The cost of frequent bug fixes and enhancements are reduced since certain fixed costs that are associated with the release are eliminated using continuous delivery.
- **Happier Team and Better Product**: Since the aim of Continuous Delivery is to make a product release less cumbersome, the team can work with greater confidence. Because of the frequency of releases, the team can work closely with users and to learn what ideas are worthwhile and what new features and functions can be implemented to help them. Continuous user feedback and new testing methodologies also increase the product's quality.

## Continuous Delivery Vs Continuous Deployment

Continuous Integration is usually the process when code changes made by different developers are integrated into the main code branch as soon as possible. It is usually done several times a day. The process ensures that code changes committed by individual developers do not divert or impact the main code branch. When combined with automated testing, it ensures that their code is dependable and can be moved into the next phase, i.e. testing or production.

Continuous deployment is somewhat similar to continuous integration. It is the process where applications can be deployed at any time to production or test environments if the current version passes all the automated unit test cases.

Continuous delivery is the methodology whereby the codebase can be deployed at any time. Apart from ensuring that your application has successfully passed all automated test cases, it also ensures that it has saved the configuration required to deploy the code in production, resulting in a faster application development lifecycle.

### How to Perform Continuous Delivery?

Let's discuss the process flow:

- The developers build their code on the local system that has all the new changes or new requirements.
- Once coding is completed, the developers need to write automated unit testing scripts that will test the code. This process is optional, however, and can be done by the testing team as well.
- A local build is executed which ensures that no breakage is occurring in the application because of the code.
- After a successful build, the developers check if any of team members or peers have checked-in anything new.
- If there are any incoming changes, they should be accepted by the developers to make sure that the uploading copy is the most recent version.
- Because of the newly merged copies, syncing the code with the main branch may cause certain conflicts.
- In case there is any conflict, they should be fixed to make sure the changes made are in sync with the main branch.
- The changes are now ready to be checked in. This process is known as a "CodeCommit."
- After the code is committed, another build of the source code is run on the integration system.
- The new and updated code is finally ready for the next stage, i.e. testing or deployment. In the next section, we shall discuss the basic checklist for continuous delivery.

## Continuous Delivery Checklist

In order to create a smooth delivery process, the following checklist should be followed before code is submitted.

- Before any changes are submitted, ensure that the current build is successful. If there are issues, fix the build before any new code is submitted.
- If the build is in the successful state, rebase the workspace to the configuration in which the build was successful.
- In the local system, build and test the code to check if any functionality is impacted because of the changes made.
- If everything goes well, check in the code.
- Allow competition of continuous integration with the new code changes.
- If somehow the build fails, stop and go back to the 3rd step in the checklist.
- If the build is successful, work can proceed on next code.

## Continuous Integration and Continuous Delivery (CI/CD) Pipeline

CI stands for Continuous Integration and CD stands for Continuous Delivery and Continuous Deployment.

A CI/CD pipeline can be easily understood as the process pathway through which developers can deliver a single unit of production-ready software. Teams choose which services they'll use to build; there is no single canonical implementation of a CI/CD pipeline.

## Using Bitbucket and AWS Codepipeline to Implement CICD Pipeline

### Source Code Control

For the best results, we will host the source code on Bitbucket as a private repository. This way, we can use its integrations with major software and services to establish triggers whenever CodeCommits are checked in.

Here we are using Bitbucket instead of AWS CodeCommit. Let's check its features.

### AWS CodeCommit Vs Bitbucket

**AWS** CodeCommit is a fully-managed source control service that makes it easy for companies to host secure and highly scalable private Git repositories. CodeCommit eliminates the need to operate a separate source control system or worry about scaling its infrastructure. Teams can use CodeCommit to securely store anything from source code to binaries, and it works seamlessly with existing Git tools.

Bitbucket gives teams one place to plan projects, collaborate on code, test and deploy, all with free private Git repositories. Teams choose Bitbucket because it has a superior Jira integration, built-in CI/CD, and is free for up to five users.

AWS CodeCommit and Bitbucket can be categorized as "Code Collaboration & Version Control" tools.

Some of the features offered by AWS CodeCommit are:

- Collaboration
- Encryption
- Access Control

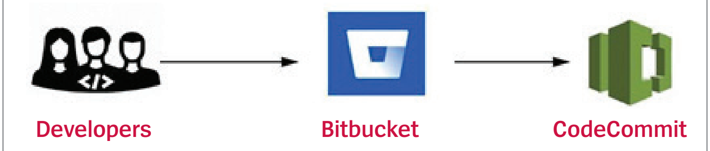
On the other hand, Bitbucket provides the following key features:

- Unlimited private repositories, charged per user
- Best-in-class Jira integration
- Built-in CI/CD

### Bitbucket Pipeline to AWS CodeCommit

In order to continue with Bitbucket as Version Control, teams use Bitbucket pipeline to sync the code from Bitbucket to AWS CodeCommit. The reason behind code syncing is that developers can use CodeCommit in the next AWS CICD pipeline. When the code is pushed or merged to Master Branch in Bitbucket, it automatically syncs to AWS CodeCommit.

From Bitbucket to AWS CodeCommit



### Step-By-Step Procedure to Sync Code From Bitbucket to AWS CodeCommit

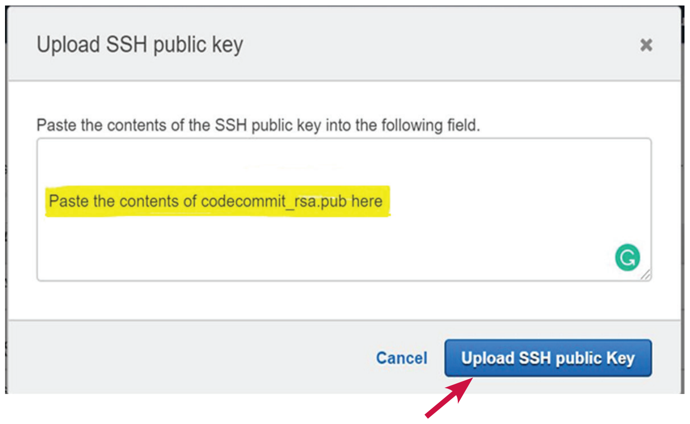
Use the following configuration steps to sync code from Bitbucket to AWS CodeCommit.

**Create a target repo:**

As a workaround to support Bitbucket with AWS Codepipeline, create a target repo in AWS CodeCommit where Bitbucket will push the code. This target repo will act as source for the pipeline in AWS Codepipeline.

**IAM User for CodeCommit:**

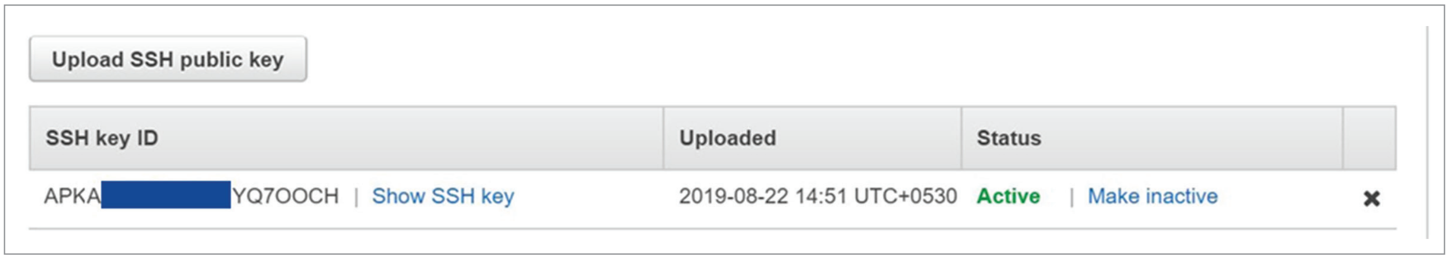
- Open the user from IAM console. Under user's section, go to Security Credentials tab
- Under SSH keys for Aws CodeCommit upload new SSH public key
- To do this open up the Git Bash terminal and add the following commands
  - `$ ssh-keygen –f ~/.ssh/codecommit_rsa`
  - And inside .ssh folder, find codecommit_rsa and codecommit_rsa.pub file
  - Open codecommit_rsa.pub file in editor and copy the contents
  - Click Upload SSH Public Key button in the popup and paste the contents of codecommit_rsa.pub file and click Upload SSHpublicKey


Upload SSH public key

Paste the contents of the SSH public key into the following field.

Paste the contents of codecommit_rsa.pub here

After uploading this, get the “SSH key ID”



Upload SSH public key

| SSH key ID | Uploaded | Status | |
|---|---|---|---|
| APKA[redacted]YQ7OOCH \| Show SSH key | 2019-08-22 14:51 UTC+0530 | Active \| Make inactive | ✖ |

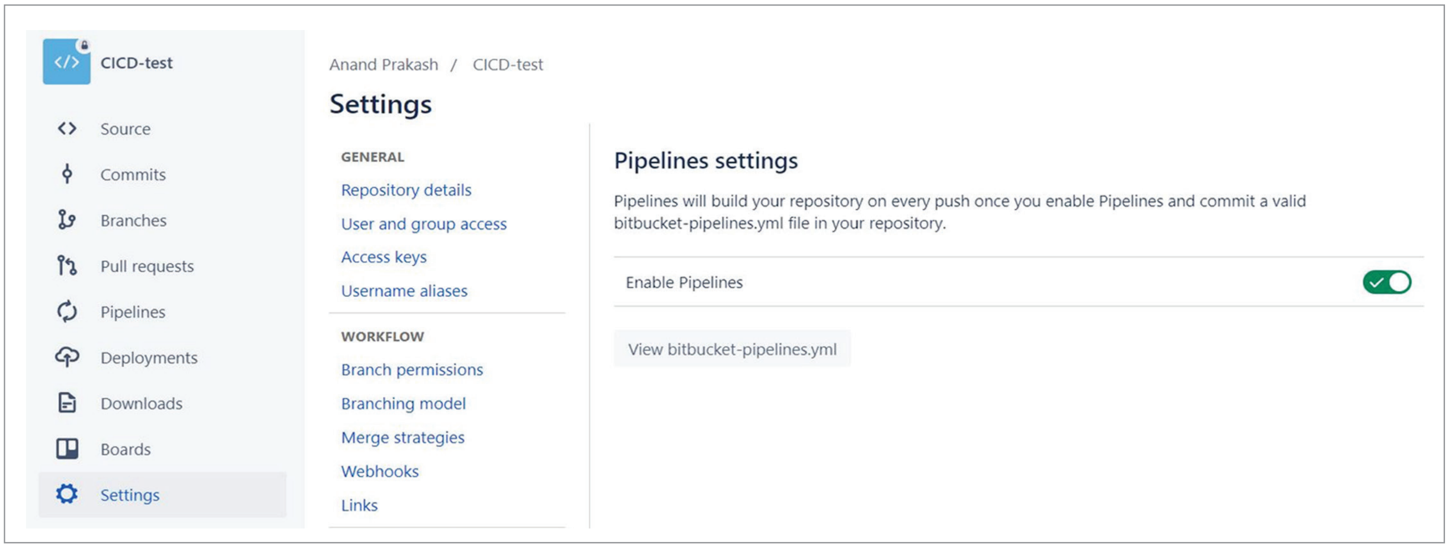
- Open the terminal and edit this file: ~/.ssh/config
- Set the values as described below and save the file

```
Host git-codecommit.*.amazonaws.com
User Your-IAM-SSH-Key-ID-Here
IdentityFile ~/Documents/.ssh/codecommit_rsa
```

- Now, create new CodeCommit repository
  - AWS Console > CodeCommit > Getting Started > Create Repository
  - Give the repository a name and description and hit the create button

## Bitbucket Pipeline Setup

- Open the repository in Bitbucket on which to setup the pipeline. [Note: Admin access is required for that repository]
- After opening the repository go to **settings** > **Pipeline Settings** > **Enable Pipeline**

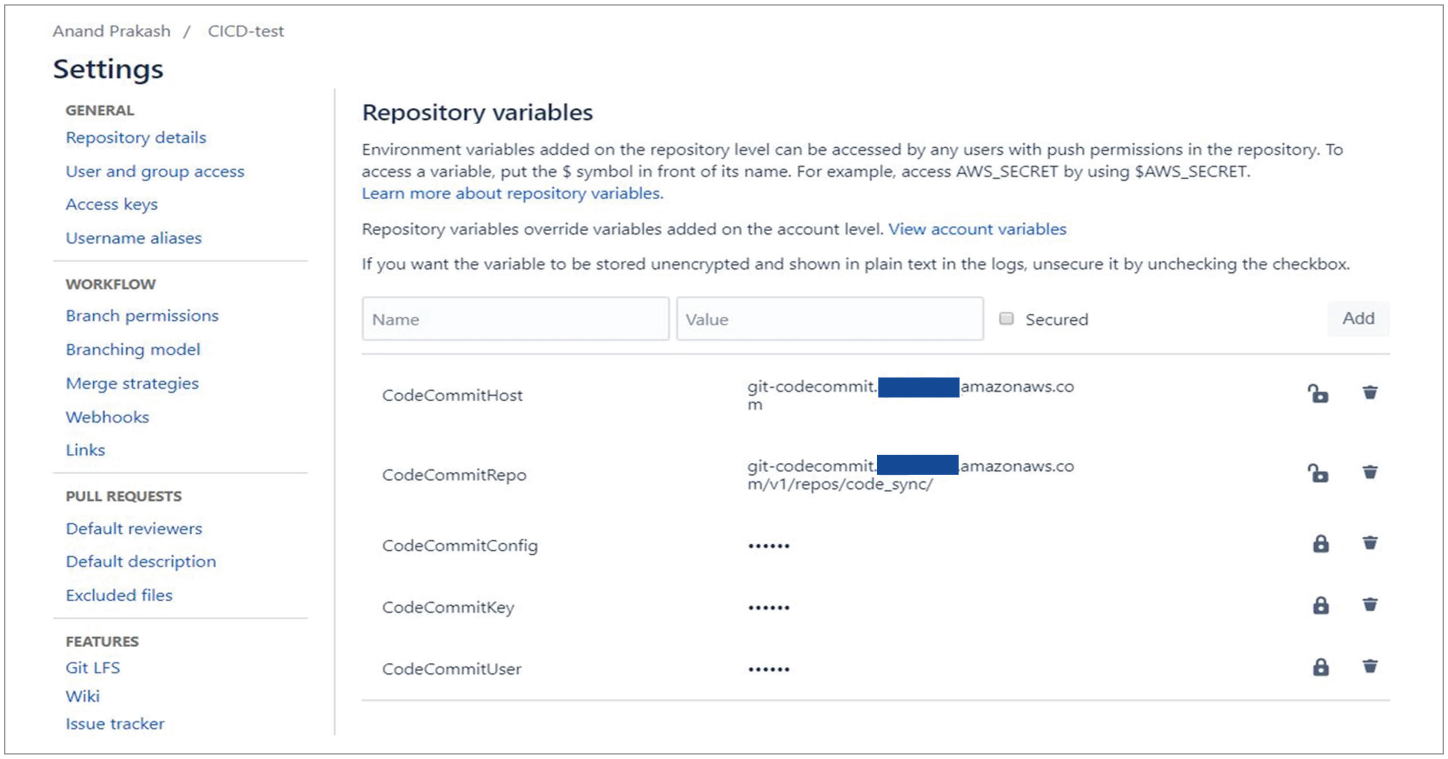


Pipelines settings

Pipelines will build your repository on every push once you enable Pipelines and commit a valid bitbucket-pipelines.yml file in your repository.

Enable Pipelines ✓

- Then go to **Pipeline** > **Settings** -> **Repository variables** and set these values:


Anand Prakash / CICD-test

Settings

Repository variables

Environment variables added on the repository level can be accessed by any users with push permissions in the repository. To access a variable, put the $ symbol in front of its name. For example, access AWS_SECRET by using $AWS_SECRET.
Learn more about repository variables.

Repository variables override variables added on the account level. View account variables

If you want the variable to be stored unencrypted and shown in plain text in the logs, unsecure it by unchecking the checkbox.

| | |
|---|---|
| CodeCommitHost | git-codecommit.[redacted]amazonaws.com |
| CodeCommitRepo | git-codecommit.[redacted]amazonaws.com/v1/repos/code_sync/ |
| CodeCommitConfig | •••••• |
| CodeCommitKey | •••••• |
| CodeCommitUser | •••••• |

## Description of Each of These Variables

- **CodeCommitConfig** -> 64-bit encoded version of the contents inside the **~/.ssh/config** file. Generate this by doing: **$ base64 ~/.ssh/config** and copy the output to the value field of **CodeCommitConfig**
- **CodeCommitHost**: the AWS CodeCommit host. Blurred part is the AWS region
- **CodeCommitKey**: 64-bit encoded version of the contents inside the private key at **~/.ssh/codecommit_rsa.** Generate this by doing: **$ base64 ~/.ssh/codecommit_rsa** and copy the output to the value field of **CodeCommitKey**
- **CodeCommitRepo**: The link of repository just created
- **CodeCommitUser**: The IAM-SSH-Key-ID

## Create Bitbucket-Pipelines.yml

The following code syncs master branch to AWS CodeCommit.

```
pipelines:
 branches:
  master:
   - step:
      script:
       - echo $CodeCommitKey > ~/.ssh/codecommit_rsa.tmp
       - base64 –decode ~/.ssh/codecommit_rsa.tmp >
        ~/.ssh/id_rsa
        - chmod 400 ~/.ssh/id_rsa
        - echo $CodeCommitConfig > ~/.ssh/config.tmp
        - base64 --decode  ~/.ssh/config.tmp > ~/.ssh/config
        - set +e
        - ssh -o StrictHostKeyChecking=no $CodeCommitHost
        - set -e
        - rm -rf .git
        - git init
        - git remote add origin ssh://$CodeCommitRepo
        - git branch
        - git status
        - git add
        - git commit -m "Sample CodeCommit"
        - git push origin master - -force
```

Note: The IAM user should have **AWSCodeCommitFullAccess, AmazonS3FullAccess** permissions.

## Implementing AWS CI/CD Pipeline

We already discussed creating a target repo and pushing the code from Bitbucket to AWS Commit. Use the same repo as our source for this pipeline.

AWS Codepipeline works seamlessly with AWS CodeBuild, a fully managed build service, so teams can easily set up a build step within a pipeline that packages code and runs unit tests. With AWS CodeBuild, developers don't need to provision, manage, or scale their own build servers. AWS CodeBuild scales continuously and processes multiple builds concurrently so that builds are not left waiting in a queue.

We must create different CodeBuilds to perform different phases in CI/CD Pipeline. For every CodeBuild, we should define a separate buildspec to perform the required operation like creating infra, creating build, running test cases, deploying, etc.

## Creating CodeBuild Project

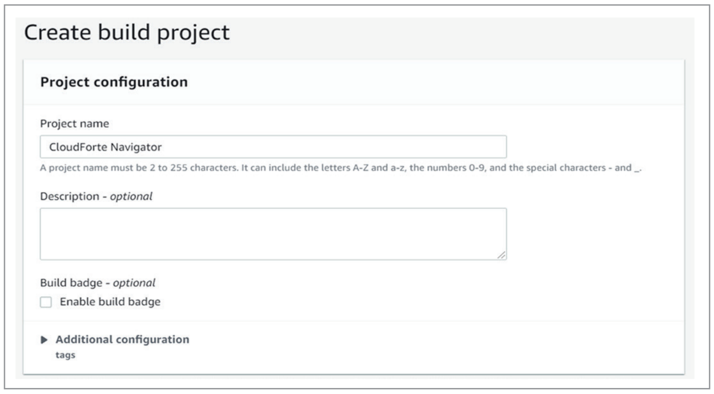
Now we will discuss how to create a CodeBuild project. (Follow the same process to create different CodeBuild projects.)

**AWS Console > CodePipeline > CodeBuild > Getting started > Create project**

Click on Create Project to see the following sections to configure per project:

Follow the screenshots of all configuration sections while creating a build. Go through them one by one to get complete information on how to create a CodeBuild project.

### Project Configuration


Create build project

Project configuration

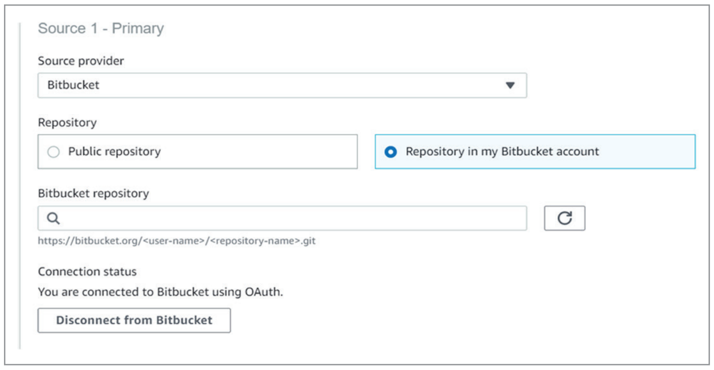
Project name
CloudForte Navigator
A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and _.

Description - *optional*

Build badge - *optional*
☐ Enable build badge

▶ Additional configuration
tags

### Source


Source 1 - Primary

Source provider
Bitbucket

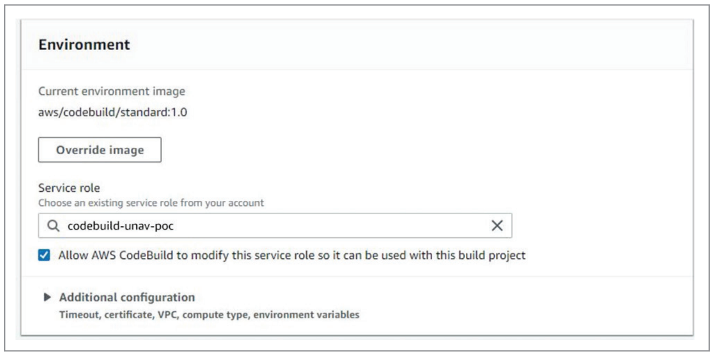
Repository
Public repository [✓ Repository in my Bitbucket account]

Bitbucket repository
https://bitbucket.org/<user-name>/<repository-name>.git

Connection status
You are connected to Bitbucket using OAuth.

Disconnect from Bitbucket

### Environment


Environment

Current environment image
aws/codebuild/standard:1.0

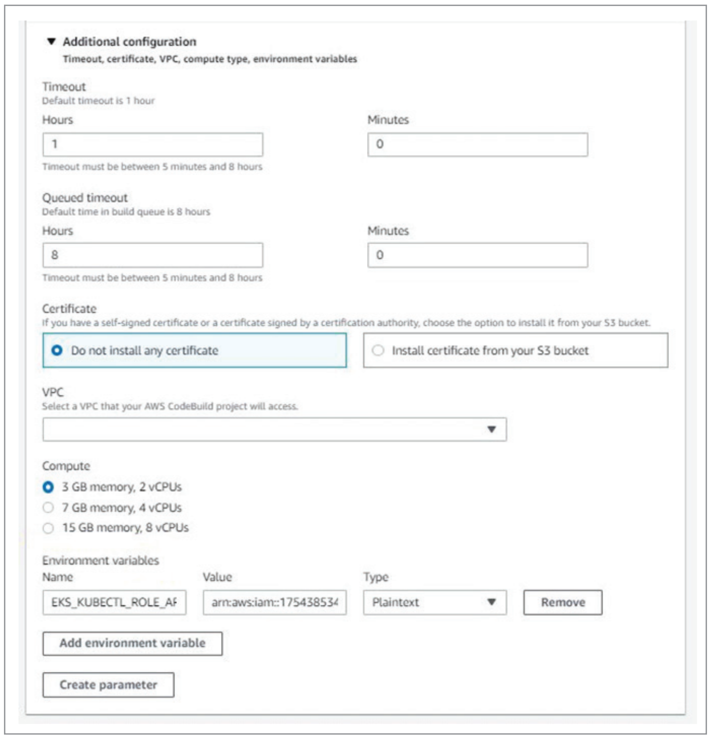
Override image

Service role
Choose an existing service role from your account
codebuild-unav-poc

☑ Allow AWS CodeBuild to modify this service role so it can be used with this build project

▶ Additional configuration
Timeout, certificate, VPC, compute type, environment variables

Click on "Additional configuration", and see the following options.


▼ Additional configuration
Timeout, certificate, VPC, compute type, environment variables

Timeout
Default timeout is 1 hour

| Hours | Minutes |
|---|---|
| 1 | 0 |

Timeout must be between 5 minutes and 8 hours

Queued timeout
Default time in build queue is 8 hours

| Hours | Minutes |
|---|---|
| 8 | 0 |

Timeout must be between 5 minutes and 8 hours

Certificate
If you have a self-signed certificate or a certificate signed by a certification authority, choose the option to install it from your S3 bucket.
[✓ Do not install any certificate] Install certificate from your S3 bucket

VPC
Select a VPC that your AWS CodeBuild project will access.

Compute
[✓ 3 GB memory, 2 vCPUs]
7 GB memory, 4 vCPUs
15 GB memory, 8 vCPUs

Environment variables

| Name | Value | Type |
|---|---|---|
| EKS_KUBECTL_ROLE_AF | arn:aws:iam::175438534 | Plaintext |

Remove

Add environment variable

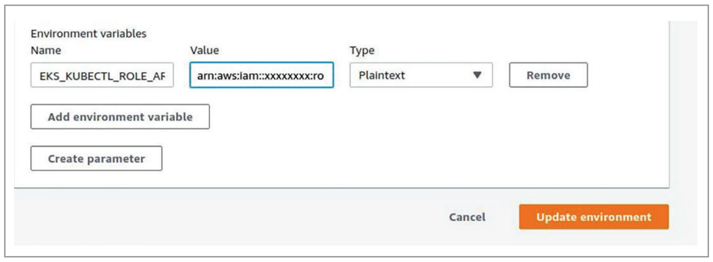
Create parameter

The following IAM Role needs to be created – EKSkubectlRole (This is a sample name only; use an appropriate custom name).

- Add trust relationships for the CodeBuild role which was created earlier. (https://docs.aws.amazon.com/directoryservice/latest/admin-guide/edit_trust.html)
- Attach EKS All policies for this role. Simply search keyword EKS to find all policies.
- Add the role ARN in environment of all CodeBuilds.

## Add Environment Variables

Depending on the environment – Dev, QA and Prod – we need to specify variables in AWS console. The following is the screenshot of variables.



Environment variables

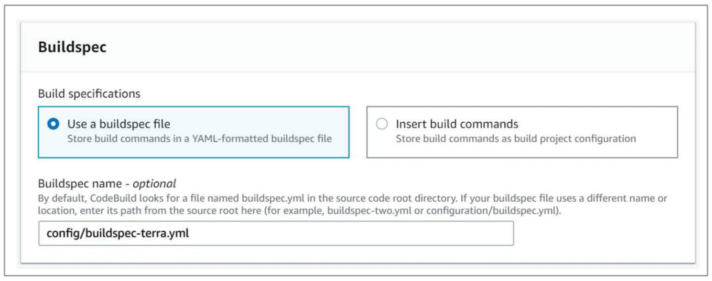
| Name | Value | Type |  |
|---|---|---|---|
| EKS_KUBECTL_ROLE_AF | arn:aws:iam::xxxxxxxx:ro | Plaintext | Remove |

To change information about environment variables the builds will use, go to **Additional configuration → Environment variables** and change the values for **Name**, **Value** and **Type**. Use **Add row** to add an environment variable.

**Things to keep in mind –**

- Make sure all the S3 Buckets are present and all files inside config/folder exist, or replace them with the newly created one.
- Make sure you created an ECR Repository.

## Buildspec



Buildspec

Build specifications

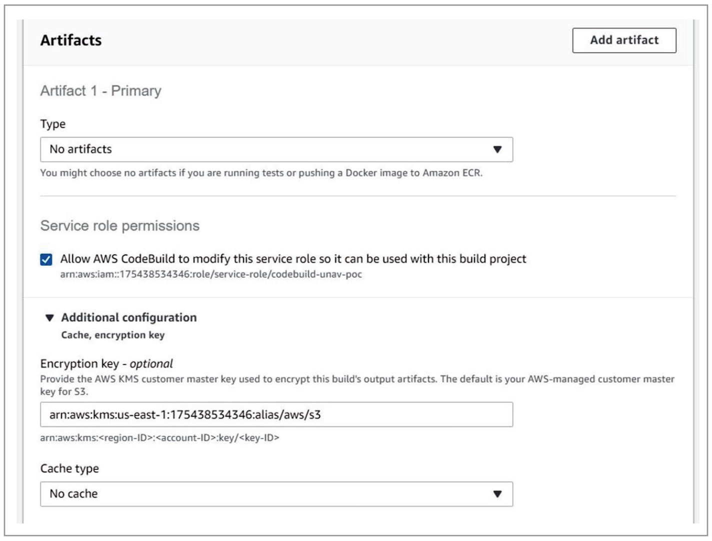
(●) Use a buildspec file
Store build commands in a YAML-formatted buildspec file

( ) Insert build commands
Store build commands as build project configuration

Buildspec name - *optional*
By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).

config/buildspec-terra.yml

## Artifacts



Artifacts

Artifact 1 - Primary

Type
No artifacts
You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.

Service role permissions

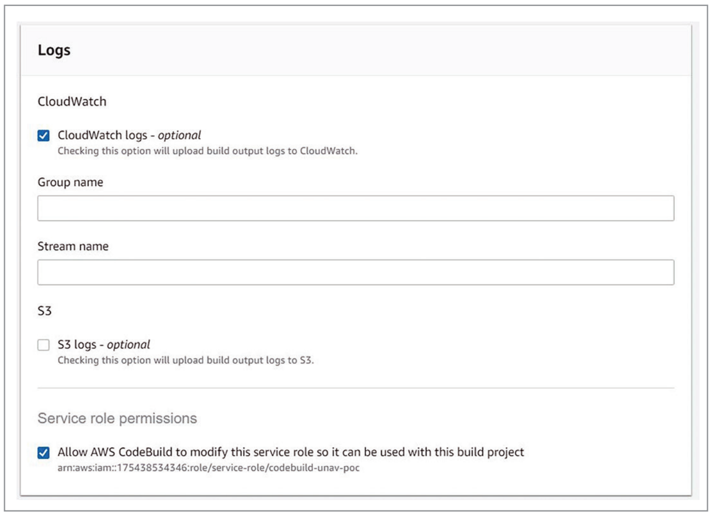
☑ Allow AWS CodeBuild to modify this service role so it can be used with this build project
arn:aws:iam::175438534346:role/service-role/codebuild-unav-poc

Additional configuration
Cache, encryption key

Encryption key - *optional*
Provide the AWS KMS customer master key used to encrypt this build's output artifacts. The default is your AWS-managed customer master key for S3.
arn:aws:kms:us-east-1:175438534346:alias/aws/s3
arn:aws:kms:<region-ID>:<account-ID>:key/<key-ID>

Cache type
No cache

## Logs



Logs

CloudWatch

☑ CloudWatch logs - *optional*
Checking this option will upload build output logs to CloudWatch.

Group name

Stream name

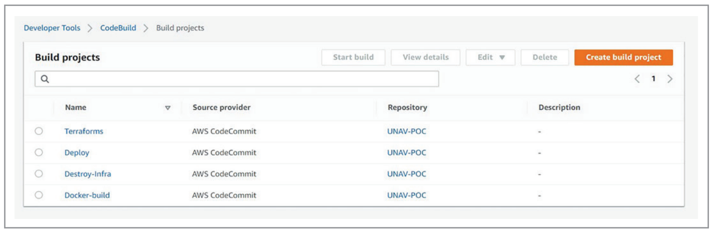
S3

☐ S3 logs - *optional*
Checking this option will upload build output logs to S3.

Service role permissions

☑ Allow AWS CodeBuild to modify this service role so it can be used with this build project
arn:aws:iam::175438534346:role/service-role/codebuild-unav-poc

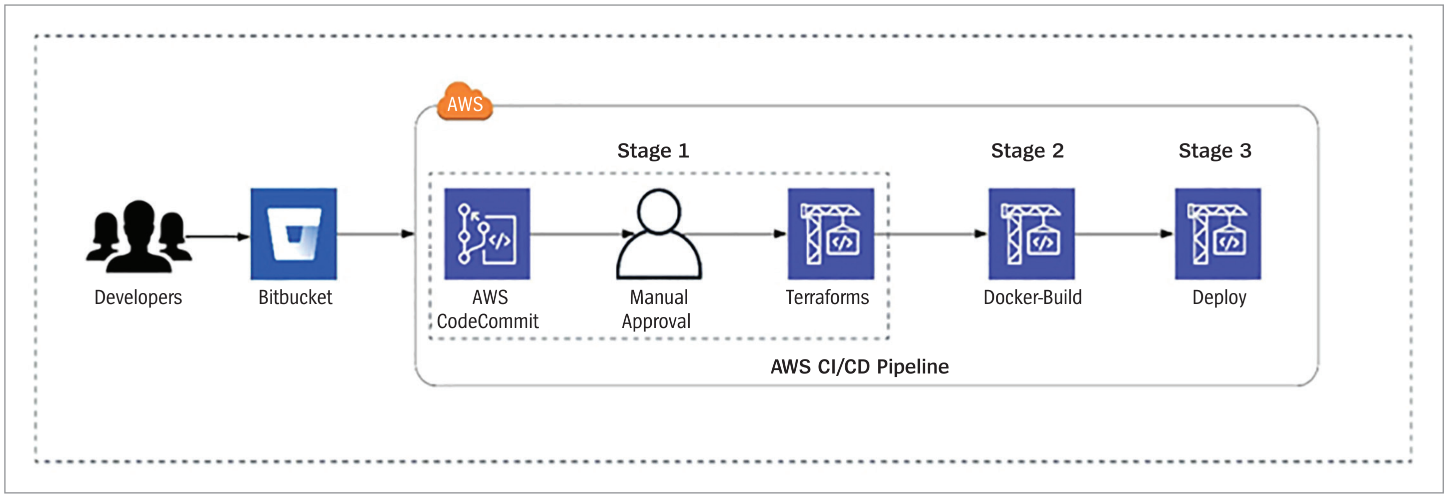
After creating the CodeBuilds, see the following CodeBuild projects. (In this example, we created only four CodeBuild projects.)



Build projects

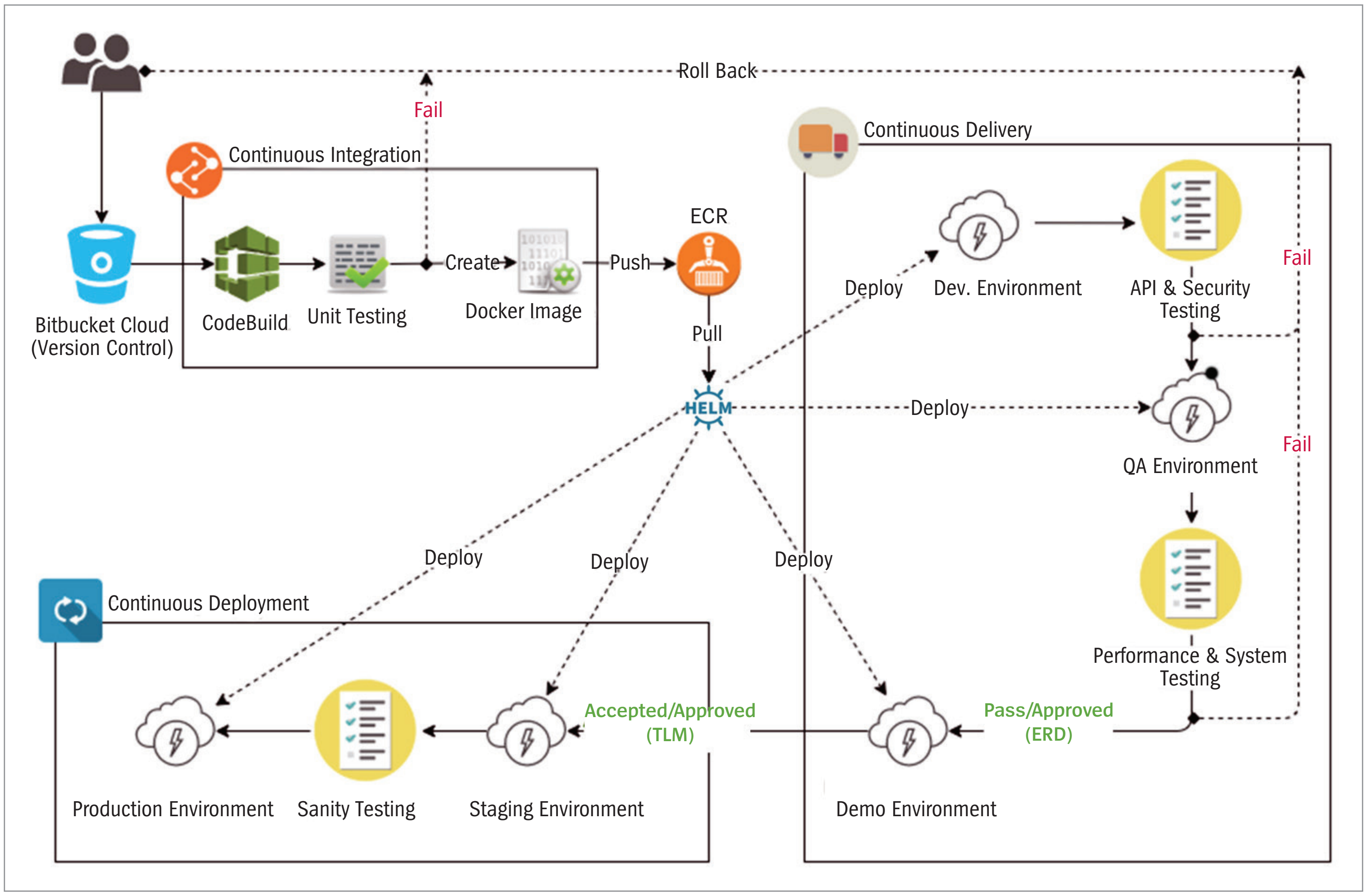
| Name | Source provider | Repository | Description |
|---|---|---|---|
| Terraforms | AWS CodeCommit | UNAV-POC | - |
| Deploy | AWS CodeCommit | UNAV-POC | - |
| Destroy-Infra | AWS CodeCommit | UNAV-POC | - |
| Docker-build | AWS CodeCommit | UNAV-POC | - |

## High Level CI/CD Pipeline



This is the overall workflow of the CI/CD functionality. First, there is a pipeline which is integrated in Bitbucket, so as the code is pushed to Master branch, it automatically pushes the code to target repo in AWS CodeCommit.

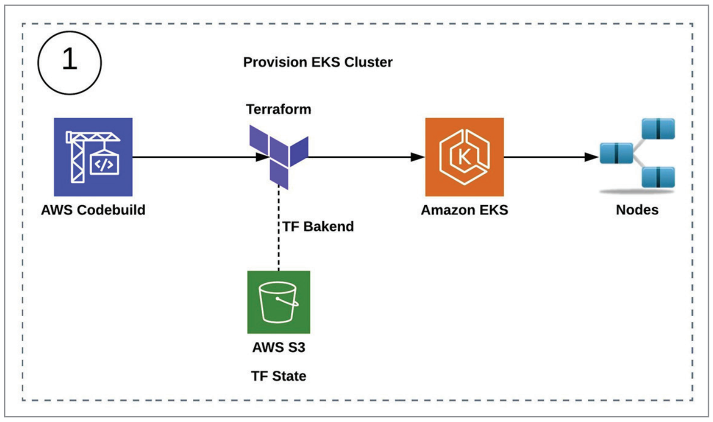
Please note that we can build a deployable image once and deploy it many times to different environments.



## AWS Pipeline Stages

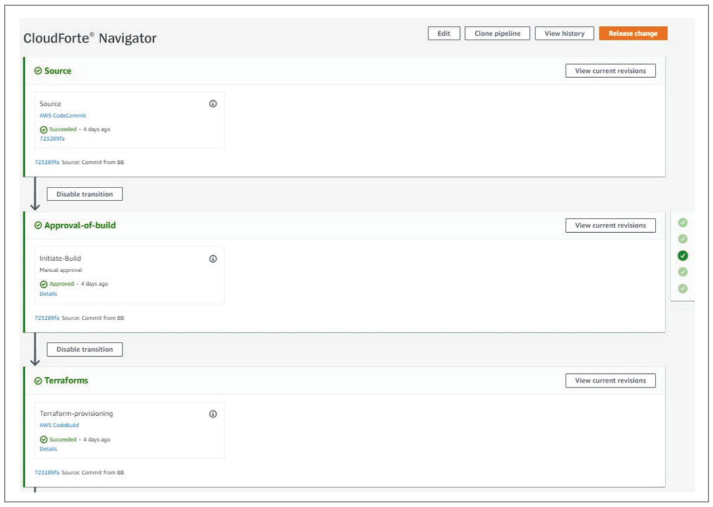
After all of the previous steps are done, setup an AWS pipeline in multiple stages.

### Stage 1: Trigger and EKS Infra Provisioning



Here, the pushed code to CodeCommit initializes and triggers the first part of the pipeline. As the first CodeBuild is triggered, it starts creating the whole infrastructure for EKS and stores the tfstate in S3 bucket. This stage takes around 10 minutes to provision.

The following is a screenshot from AWS console of this stage.



CloudForte® Navigator

Source

Source
AWS CodeCommit
Succeeded - 4 days ago
723289fa

723289fa Source: Commit from BB

Approval-of-build

Initiate-Build
Manual approval
Approved - 4 days ago
Details

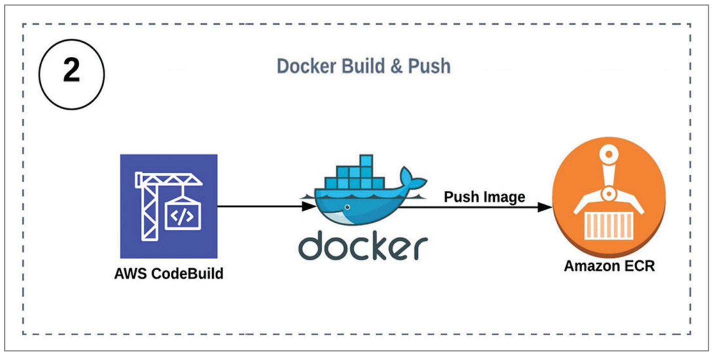
723289fa Source: Commit from BB

Terraforms

Terraform-provisioning
AWS CodeBuild
Succeeded - 4 days ago
Details

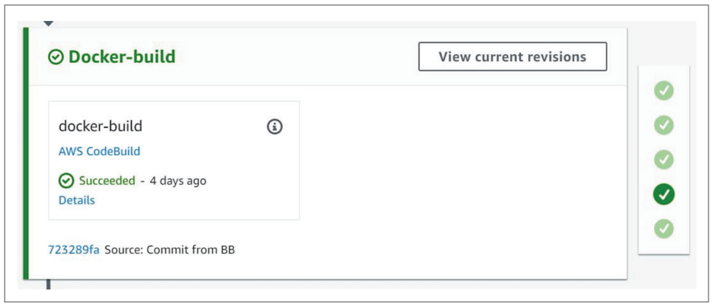
723289fa Source: Commit from BB

### Stage 2: Build Docker Image and Push to ECR



After all test cases have successfully passed, it goes to the next stage, where it builds the docker image of the application and pushes the same to ECR. This stage takes around 5-7 mins.

Following is a screenshot from AWS console of this stage.

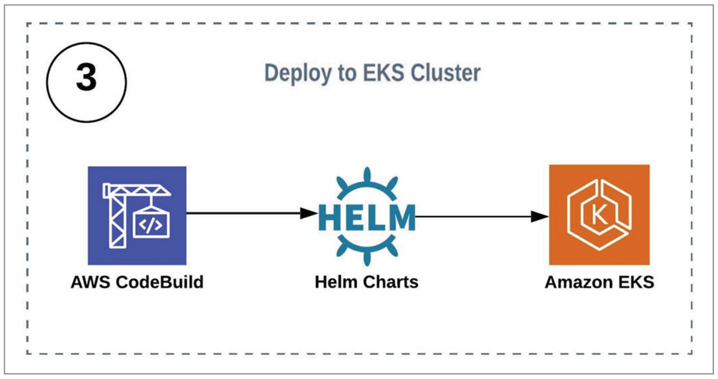


Docker-build

docker-build
AWS CodeBuild
Succeeded - 4 days ago
Details

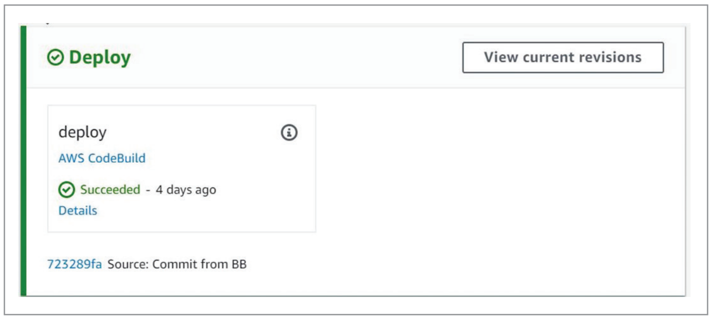
723289fa Source: Commit from BB

### Stage 3: Deploy to EKS



Next, the pushed image from ECR, gets deployed to EKS Cluster via Helm charts, and the URL is provided to test the application. This stage takes around 1-2 minutes.

The following is a screenshot from AWS console of this stage.



Deploy

deploy
AWS CodeBuild
Succeeded - 4 days ago
Details

723289fa Source: Commit from BB

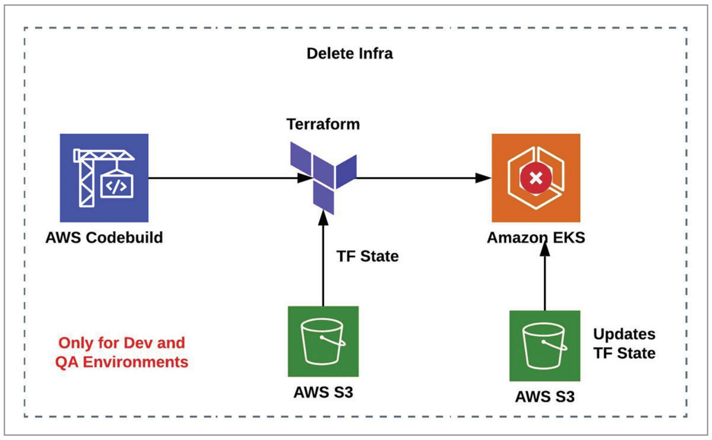
## Delete Infra

Once all testing and build is certified, the created infra is no longer required. This stage completely destroys the infrastructure to save costs.

This stage is not included in the pipeline and CodeBuild must be triggered manually with the following command:

**AWS Console > CodePipeline > CodeBuild > Destroy-Infra > Start build**

Click on Start build. It asks for the repository. Select "master repository" and trigger the build.



This stage takes around 5-8 minutes.

We realized many benefits by adopting this practice. We brought down the 'new feature' deployment time to less than 30 minutes, compared to two weeks. This helped our team focus on core activities and allowed us to provision another production-like environment in less than half a day compared to two weeks. We also achieved zero known security vulnerability in our latest deployment.

## References

https://semaphoreci.com/blog/cicd-pipeline

https://aws.amazon.com/getting-started/projects/set-up-ci-cd-pipeline/

https://docs.aws.amazon.com/directoryservice/latest/admin-guide/edit_trust.html

https://docs.aws.amazon.com/codecommit/latest/userguide/how-to-create-repository.htm
l#how-to-create-repository-console

**For more information visit www.unisys.com/cloudforte®**

For more information visit www.unisys.com